Performance Simulation for Unit-Memory Convolutional Codes With Byte-Oriented Viterbi Decoding Algorithm

Q. D. Vo
Communications Systems Research Section

This article describes a software package developed to simulate the performance of the byte-oriented Viterbi decoding algorithm for unit-memory (UM) codes on both 3-bit and 4-bit quantized AWGN channels. The simulation was shown to require negligible memory and less time than that for the RTMBEP algorithm, although they both provide similar performance in terms of symbol-error probability. This makes it possible to compute the symbol-error probability of large codes and to determine the signal-to-noise ratio required to achieve a bit error rate (BER) of 10^{-6} for corresponding concatenated systems. A (7, 10/48) UM code, 10-bit Reed-Solomon code combination was found to achieve the required BER at 1.08 dB for a 3-bit quantized channel and at 0.91 dB for a 4-bit quantized channel.

I. Introduction

A general $(l_0, k_0/n_0)$ unit-memory (UM) convolutional encoder is shown in Fig. 1. Let $\mathbf{a}_t$ be the k_0-bit byte of input to be encoded at time t, $\hat{\mathbf{a}}_{t-1}$ be the l_0-bit byte of delayed input, and $\mathbf{b}_t$ be the corresponding n_0-bit byte of encoded output. Let G_0 and G_1 be encoding matrices with dimensions $k_0 \times n_0$ and $l_0 \times n_0$ respectively, then the encoding equation may be written as

$$\mathbf{b}_t = \mathbf{a}_t G_0 + \hat{\mathbf{a}}_{t-1} G_1; \qquad t = 1, 2, \cdots$$

There are two different decoding algorithms that exhibit similar performance: the RTMBEP and the byte-oriented Viterbi. The RTMBEP decoding rule, which has been previously simulated (Ref. 1), has as its estimate $\mathbf{a}_t^0$ the value of $\mathbf{a}_t$ that maximizes $P(\mathbf{a}_t | \mathbf{r}_{[1,t+\Delta]})$ where $\mathbf{r}[1, t + \Delta]$ is the observed sequence with delay Δ. To speed up the simulation, we set up probability matrices $P(\mathbf{r}_{t+i} | \mathbf{b}_{t+i})$, where $i = 0, 1, \cdots, \Delta$. As a result, the required memory is at least $(\Delta + 1)2^{l_0+k_0}$, which is quite large for big codes. For example, for $l_0 = 9$, $k_0 = 10$, and $\Delta = 8$, at least 4,718,592 real numbers are needed. On the other hand, the simulation for the byte-oriented Viterbi decoding algorithm requires practically no memory (too small to count) since it does not have to store the matrix $P(\mathbf{r}_t | \mathbf{b}_t)$. Furthermore, the Viterbi algorithm itself is much simpler and hence runs faster than the RTMBEP algorithm.

II. Byte-Oriented Viterbi Decoding Algorithm

The byte-oriented Viterbi decoding rule chooses its estimated sequence $\mathbf{a}_1^0, \mathbf{a}_2^0, \cdots, \mathbf{a}_I^0$ to be the value of $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_I$, which maximizes

$$P(\mathbf{a}_1, \cdots, \mathbf{a}_I | \mathbf{r}_1, \cdots, \mathbf{r}_I) = \frac{P(\mathbf{r}_1, \cdots, \mathbf{r}_I | \mathbf{a}_1, \cdots, \mathbf{a}_I) P(\mathbf{a}_1, \cdots, \mathbf{a}_I)}{P(\mathbf{r}_1, \cdots, \mathbf{r}_I)}$$

Here we assume that all information sequences are equally likely (i.e., $P(\mathbf{a}_t) = 2^{-k_0}$) so that the algorithm is the same as maximizing

$$P(\mathbf{r}_1, \cdots, \mathbf{r}_I | \mathbf{a}_1, \cdots, \mathbf{a}_I)$$

or

$$P(\mathbf{r}_1, \cdots, \mathbf{r}_I | \mathbf{b}_1, \cdots, \mathbf{b}_I(\mathbf{a}_1, \cdots, \mathbf{a}_I))$$

A recursive method is developed as follows. Let $\mathbf{a}_t = \tilde{\mathbf{a}}_t, \hat{\mathbf{a}}_t$ where the comma denotes concatenation of $(k_0 - l_0)$-bit byte $\tilde{\mathbf{a}}_t$ with l_0-bit byte $\hat{\mathbf{a}}_t$. $\hat{\mathbf{a}}_t$ is called the state at time t since it affects the output at time $t + 1$. Let

$$f(\hat{\mathbf{a}}_{t-1}) = \max_{\mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}} P(\mathbf{r}_1, \cdots, \mathbf{r}_{t-1} | \mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}, \hat{\mathbf{a}}_{t-1})$$

Then

$$f(\hat{\mathbf{a}}_t) = \max_{\mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}, \hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t} P(\mathbf{r}_1, \cdots, \mathbf{r}_t | \mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}, \hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t, \hat{\mathbf{a}}_t)$$

$$= \max_{\mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}, \hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t} \left\{ \begin{array}{l} P(\mathbf{r}_t | \mathbf{r}_1, \cdots, \mathbf{r}_{t-1}, \mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}, \hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t, \hat{\mathbf{a}}_t) \\ \times P(\mathbf{r}_1, \cdots, \mathbf{r}_{t-1} | \mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}, \hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t, \hat{\mathbf{a}}_t) \end{array} \right\}$$

Since the code has unit memory we can write

$$f(\hat{\mathbf{a}}_t) = \max_{\hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t} \left\{ P(\mathbf{r}_t | \hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t, \hat{\mathbf{a}}_t) \left\{ \max_{\mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}} P(\mathbf{r}_1, \cdots, \mathbf{r}_{t-1} | \mathbf{a}_1, \cdots, \tilde{\mathbf{a}}_{t-1}, \hat{\mathbf{a}}_{t-1}) \right\} \right\}$$

$$= \max_{\hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t} \left\{ P(\mathbf{r}_t | \mathbf{b}_t(\hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t, \hat{\mathbf{a}}_t)) f(\hat{\mathbf{a}}_{t-1}) \right\}$$

The process can be described by a trellis diagram (Fig. 2). At time t we have to compute and store the metric at each state (i.e., $f(\hat{\mathbf{a}}_t)$). Also, we need to store the corresponding optimal path leading to that state. The performance simulation software package for the byte-oriented Viterbi decoding algorithm is summarized as follows:

(1) Initialization ($t = 0$)

(a) Set up coder matrix that gives $\mathbf{b}_t$ for each $\mathbf{a}_t$ and $\hat{\mathbf{a}}_{t-1}$.

(b) Set up 3-bit quantized AWGN channel probability matrix.

(c) Set

$$f(\hat{\mathbf{a}}_0 = \mathbf{0}) = 1 \text{ and } f(\hat{\mathbf{a}}_0 \neq \mathbf{0}) = 0$$

$$\text{IERR}(\hat{\mathbf{a}}_0) = 0$$

Note that since we are interested just in symbol error probability, in the simulation we need to store only the accumulated number of errors of the optimal path that leads to a particular state (i.e., IERR $(\hat{\mathbf{a}}_t)$).

(2) At time t (main loop; $t = 1, 2, \cdots$), the following steps are taken:

(a) Simulate current observed byte $\mathbf{r}_t$.

(b) For each state $\hat{\mathbf{a}}_t$, compute

$$f(\hat{\mathbf{a}}_t) = \max_{\hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t} \left\{ P(\mathbf{r}_t \mid \mathbf{b}_t(\hat{\mathbf{a}}_{t-1}, \tilde{\mathbf{a}}_t, \hat{\mathbf{a}}_t)) f(\hat{\mathbf{a}}_{t-1}) \right\}$$

and count the corresponding number of errors IERR $(\hat{\mathbf{a}}_t)$. Since $f(\hat{\mathbf{a}}_t)$ will be very small after many iterations, it is necessary to normalize it:

$$f(\hat{\mathbf{a}}_t) = f(\hat{\mathbf{a}}_t)/f(\mathbf{0})$$

(3) Finally, the estimate $\mathbf{a}_1^0, \mathbf{a}_2^0, \cdots, \mathbf{a}_I^0$ is chosen to be the path that leads to the state $\hat{\mathbf{a}}_I^0$ such that

$$f(\hat{\mathbf{a}}_I^0) \geq f(\hat{\mathbf{a}}_I), \quad \text{for all } \hat{\mathbf{a}}_I\text{'s}$$

with its corresponding number of errors IERR $(\hat{\mathbf{a}}_I^0)$.

III. Performance

The above simulation software package requires little memory (in the order of 2^{k_0}) compared to the one for RTMBEP algorithm (in the order of $2^{k_0+l_0}$). This occurs since in the RTMBEP algorithm we need to store the probability matrices $P(\mathbf{r}_{t+i} \mid \mathbf{b}_{t+i})$ where $i = 0, 1, \cdots, \Delta$ so that not all of these have to be recalculated in the next iteration. Even so, the RTMBEP algorithm is still slower due to complicated recursive procedures (see Ref. 1). The Viterbi decoder runs about four times faster for small codes ($k_0 = 4$) and about twice as fast for big codes ($k_0 = 9$). Amazingly enough, with all these advantages, the Viterbi algorithm still achieves similar performance. For comparison, results based on 8000-byte decoding simulation are shown in Table 1 for a (4,4/8) code and a (6,6/30) code. The Viterbi algorithm simulation is run for a (6,9/36) code and a (7,10/48) code found by Pil Lee. The symbol-error probabilities based on 4000-byte decoding simulation for the (6,9/36) code and 2000-byte decoding simulation for the (7,10/48) code are given in Table 2 and plotted in Fig. 3 for both 3-bit and 4-bit quantized AWGN channels (see the Appendix). These codes are concatenated with various matching symbol-size Reed-Solomon codes. The required E_b/N_0 (i.e., outer code signal-to-noise ratio) to achieve a bit-error-rate (BER) of 10^{-6} is shown in Table 3 and plotted in Fig. 4. With 4-bit channel output quantization, the (7,10/48) unit-memory code, (1023, 927) Reed-Solomon code combination requires only 0.91 dB in E_b/N_0. This represents an improvement of 1.62 dB over the proposed NASA standard (i.e., (7,1/2) convolutional code, (255,223) Reed-Solomon code combination).

IV. Conclusion

A software package was developed to simulate the performance of the byte-oriented Viterbi decoding algorithm for unit-memory codes. This simulation requires negligible memory compared to that for the RTMBEP algorithm. It also runs faster because of its simplicity. As a result, it is possible to determine the symbol-error probability for large byte-oriented codes. Then the required E_b/N_0 to achieve a BER of 10^{-6} can be evaluated for concatenated systems. A (7,10/48) code, (1023,927) Reed-Solomon code combination is found to achieve the required BER at 0.91 dB, which is a 1.62-dB improvement over the proposed NASA standard.

Acknowledgment

The author would like to thank Pil J. Lee who provided extensive code search for unit memory codes and Dr. James Lesh for his suggestions.

Reference

1. Vo, Q. D., "Simulations for Full Unit-Memory and Partial Unit-Memory Convolutional Codes with Real-Time Minimal-Byte-Error Probability Decoding Algorithm," *The Telecommunications and Data Acquisition Progress Report 42-76*, pp. 77-81, Jet Propulsion Laboratory, Pasadena, California, February 15, 1984.

Table 1. Performance comparison between Viterbi and RTMBEP decoding algorithms

Code	G_0	G_1	Viterbi decoding				RTMBEP decoding			
(4,4/8)	87 4B 2D 1E	8B E2 B8 D1	E'_b/N_0 (dB) P_s	1.5 0.0313	2.0 0.0157	2.5 0.0063	E'_b/N_0 (dB) P_s	1.5 0.0317	2.0 0.0154	2.5 0.0065
(6,6/30)	20FBAC1C 107DD60E 08BEE307 04DD71A3 02EEB0F1 01F75878	0F14B4C1 1E296982 3C52C344 39A19688 33472D10 278A5A60	E'_b/N_0 (dB) P_s	0.75 0.0252	1.00 0.0122	1.25 0.0077	E'_b/N_0 (dB) P_s	0.75 0.0246	1.00 0.0125	1.25 0.0079

Table 2. Simulated symbol-error probability for a (6, 9/36) code and a (7, 10/48) code–Viterbi decoding

Code	G_0	G_1	Symbol-error probability, P_s			
(6,9/36)	FFFFF0000 FFC00FFE0 F83E0FC1F E4210C3D8 07398B018 D71062816 909EC4234 6064DA924 6A0956DB0	C144DAA24 92168221E 973860692 072155018 60A50EACA 7008D69B1	E'_b/N_0 (dB) 4-bit channel 3-bit channel	0.3 0.0765 0.0841	0.5 0.0567 0.0605	0.7 0.0328 0.0446
(7,10/48)	FFFFFFF00000 FFFC000FFFC0 FE03F80FE03F C183870E183C 3D7B44C9DF22 BAE2B7AFB4FB 415CF745D496 3846FE7B6C28 B101CDE50AB4 73F328165182	AA84C7D08C3B A4DF8474F71D C11A5A2916B4 A65295EC8A17 DE156BCAEA0B 7EE41D2591E3 486C6ECAD964	E'_b/N_0 (dB) 4-bit channel 3-bit channel	0.0 0.0850 0.0965	0.25 0.0460 0.0590	0.50 0.0245 0.0380

Table 3. Required E_b/N_0 to achieve a BER of 10^{-6}

Code	(6,9/36)				(7,10/48)			
R-S code rate	399/511	415/511	431/511	447/511	831/1023	863/1023	895/1023	927/1023
Required P_s	0.06214	0.05080	0.03986	0.02942	0.06143	0.04914	0.03722	0.02580
Required E'_b/N_0 (dB) 3-bit	0.482	0.618	0.77	0.987	0.23	0.354	0.512	0.716
Required E'_b/N_0 (dB) 4-bit		0.557	0.65	0.724		0.222	0.335	0.482
Required E_b/N_0 (dB) 3-bit	1.56	1.52	1.51	1.57	1.13	1.09	1.09	1.14
Required E_b/N_0 (dB) 4-bit		1.46	1.39	1.31		0.96	0.92	0.91

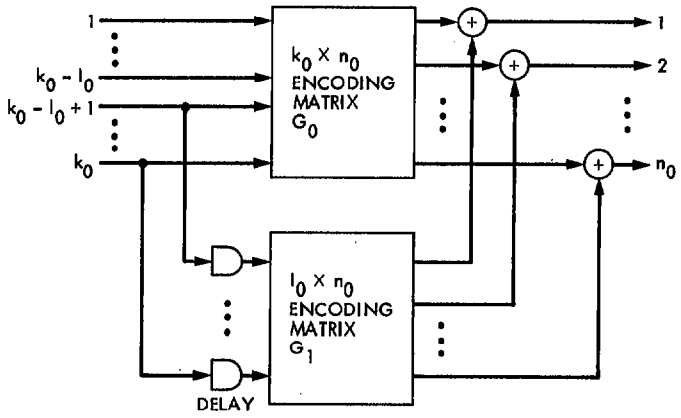

Fig. 1. A general (l_o, k_o/n_o) unit-memory convolutional encoder

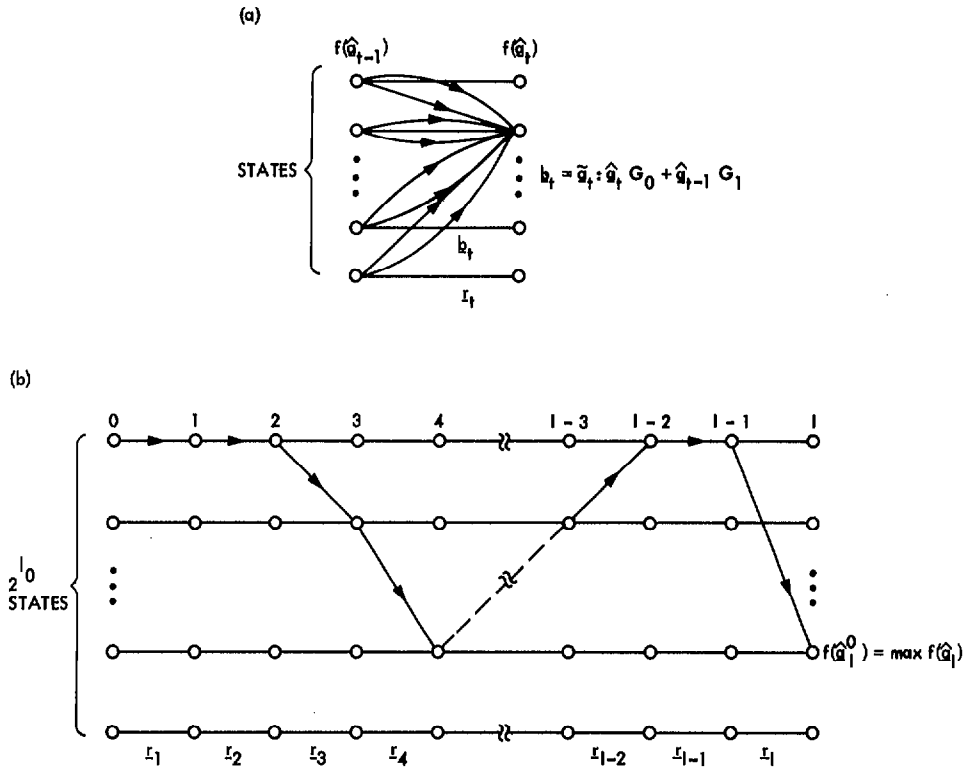

Fig. 2. Trellis diagram: (a) byte-oriented Viterbi decoding algorithm; (b) optimal path tracing

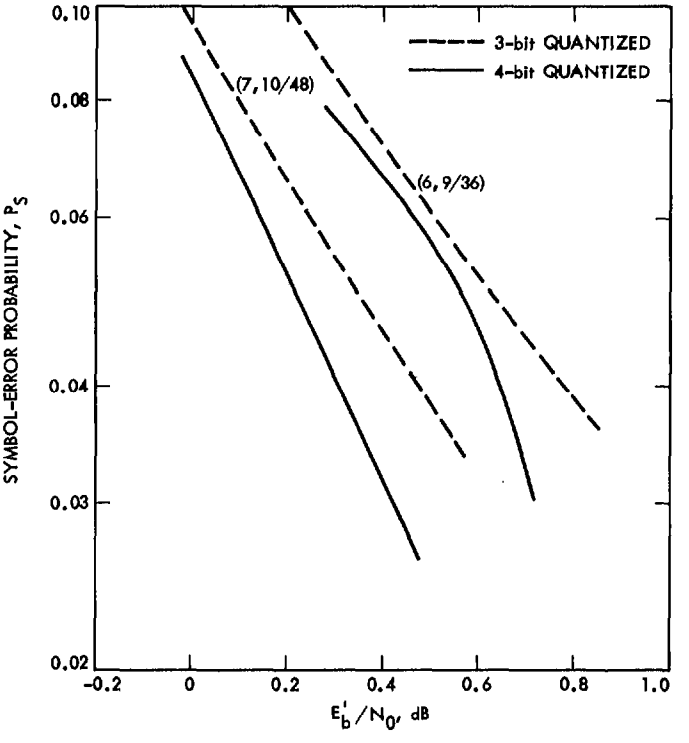

Fig. 3. Simulated symbol-error probability for a (6, 9/36) code and a (7, 10/48) code

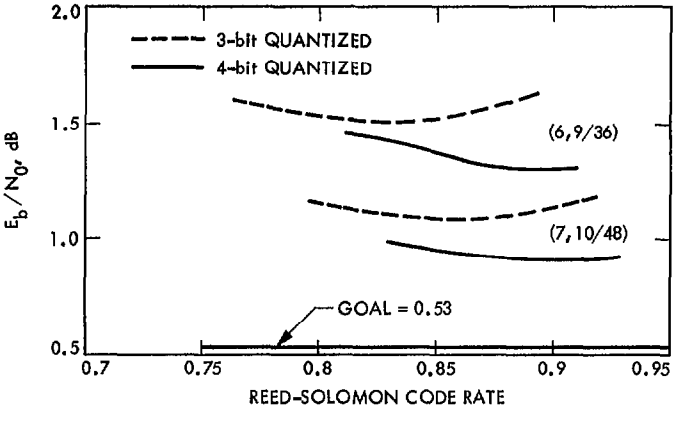

Fig. 4. Required E_b/N_0 to achieve a BER of 10^{-6}

Appendix
4-bit vs 3-bit Channel Output Quantization

The quantization of the output to one of J levels simply transforms the AWGN channel to a finite-input, finite-output alphabet channel (Ref. A-1). For our case, a biphase modulated AWGN channel with output quantized to eight levels (3-bit quantizer) is shown in Fig. A-1. The channel conditional probabilities can be computed as follows:

$$P(1|0) = Q(3a - x)$$

$$P(M|0) = Q((4 - M)a - x) - Q((5 - M)a - x); \qquad M = 2, \cdots, 7$$

$$P(8|0) = 1 - Q(-3a - x)$$

$$P(M|1) = P(9 - M|0)$$

where

$$a = \text{quantizer step size}$$

$$x = \sqrt{2E_s/N_0}$$

The quantizer step size is chosen to maximize the Bhattacharyya distance:

$$d = -\ln \sum_{M=1}^{8} \sqrt{P(M|0)\,P(M|1)}$$

From this, the channel cutoff rate R_0 can be easily computed:

$$R_0 = \max_q \left\{ -\ln \sum_y \left[\sum_x q(x) \sqrt{P(y|x)} \right]^2 \right\}$$

$$= \max_q \left\{ -\ln \sum_x \sum_{x'} q(x)\, q(x') \sum_y \sqrt{P(y|x)\,P(y|x')} \right\}$$

For binary input, this becomes

$$R_0 = -\ln \left[\frac{1}{2} + \frac{1}{2} \sum_y \sqrt{P(y|0)\,P(y|1)} \right] = -\ln \left(\frac{1 + e^{-d}}{2} \right)$$

The 4-bit quantized channel model is shown in Fig. A-2 with the channel conditional probabilities given by

$$P(1|0) = Q(7a - x)$$

$$P(M|0) = Q((8 - M)a - x) - Q((9 - M)a - x); \qquad M = 2, \cdots, 15$$

$$P(16|0) = 1 - Q(-7a - x)$$

$$P(M|1) = P(17 - M|0)$$

where the quantizer step size a is chosen to maximize

$$d = -\ln \sum_{M=1}^{16} \sqrt{P(M|0)\,P(M|1)}$$

The cutoff rate is plotted for both channels over the interested range of E_s/N_0 in Fig. A-3. We see that, to achieve the same cutoff rate, we can save approximately 0.11 dB (in the range around $E_s/N_0 = -6$ dB) in required signal-to-noise ratio by using a 4-bit quantizer instead of a 3-bit quantizer. This fact was originally suggested by Pil Lee (personal communication).

Reference

A-1. Viterbi, A. J., and J. K. Omura, *Principles of Digital Communication and Coding*, pp. 78-82, McGraw-Hill, New York, N.Y., 1979.

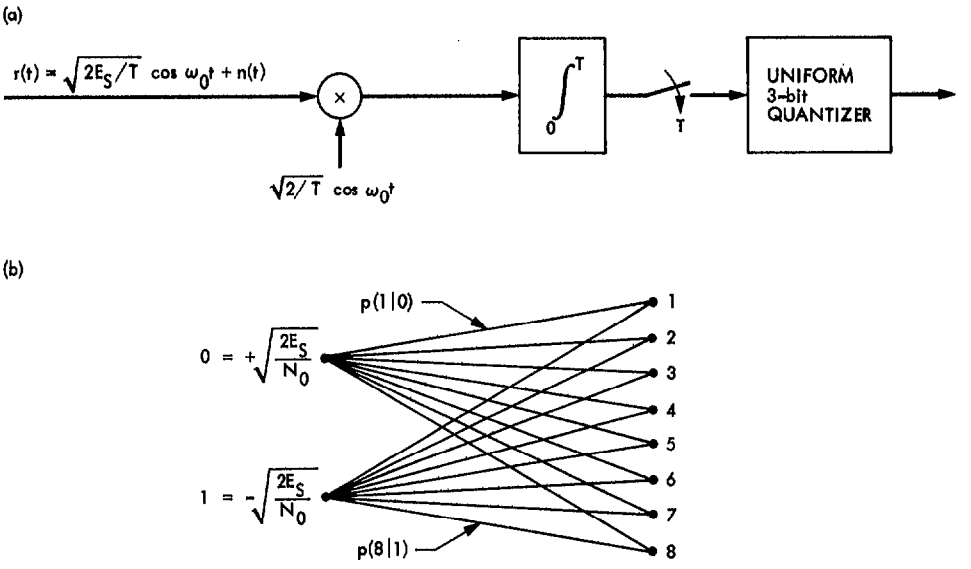

Fig. A-1. 3-bit quantized demodulator and channel model: (a) demodulator for BPSK signals; (b) channel model

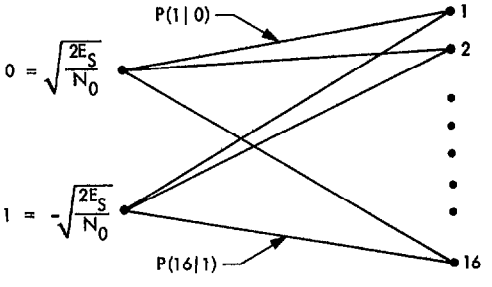

Fig. A-2. 4-bit quantized channel model

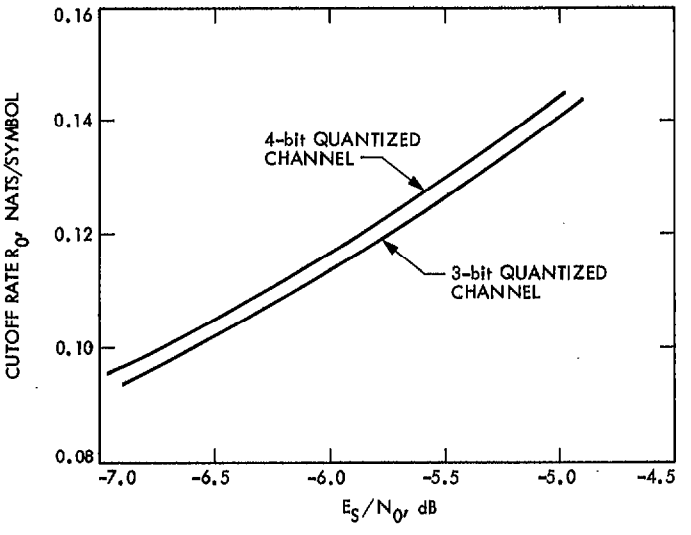

Fig. A-3. Channel cutoff rate